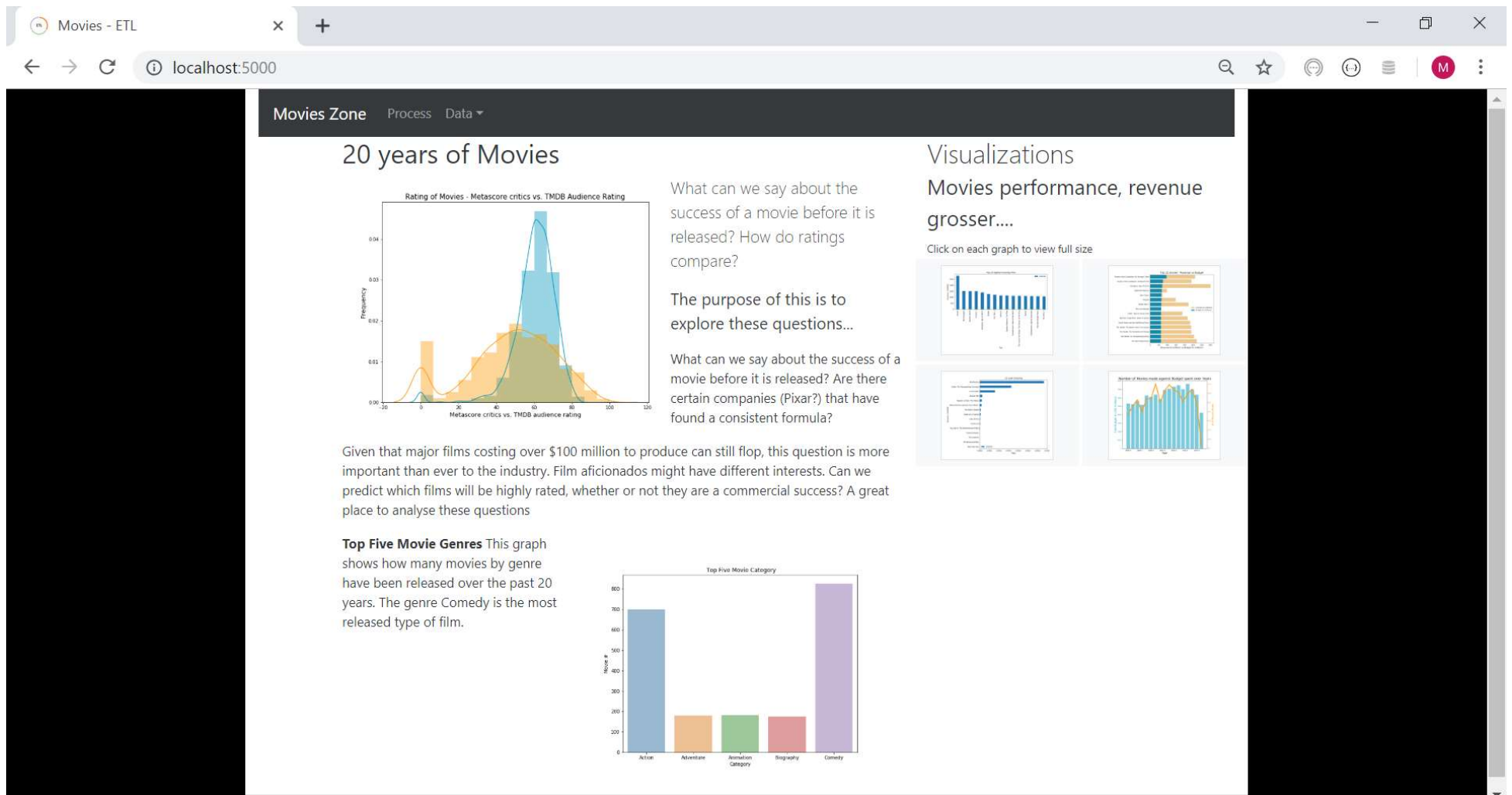
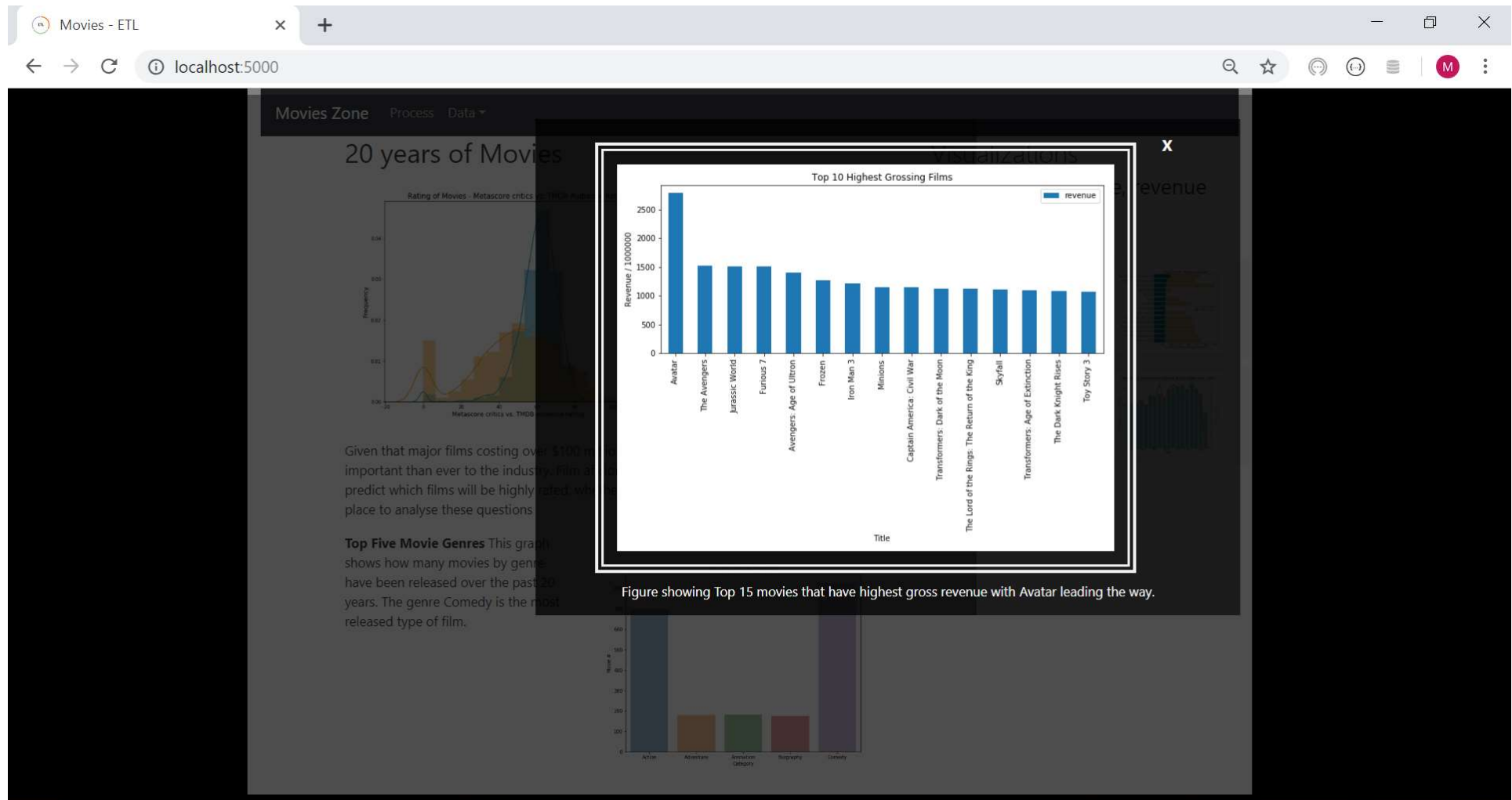


# Index page with Visualization and links



# Image Full size pop-up when clicked



# Process followed by our team in this project

Movies - ETL - Process Followed

localhost:5000/process

Movies Zone

Process

Data

Process Scope & Steps

In this project, we will

- Define objectives (Why, what and how).
- Define Sources of Data and Destination where it will be stored
- Define Data Extraction, Transformation and Load process
- Perform Data Extraction, Transformation and Load
- Data Analysis & Visualization
- Publish results on web using Flask

EXTRACTION

EXPLORE DATA

TRANSFORM

LOAD DATA

VISUALIZE

PUBLISH

Data obtained from 2 data sources

- TMDB- 5000.csv from Kaggle
- JSON from OMDB API

Python script created for data extraction. Libraries used will be,

- Pandas – extraction
- Requests – API

Explore and Identify,

- Datasets info – size, columns and datatypes
- Categorical / non-categorical columns
- Primary column(s) for merging
- Data Issues like, missing values
- Required columns for analytics

Using Python,

- Handle missing data
- Apply target formatting
- Rename Columns
- Filter and Group
- Derive new fields
- Merge the 2 datasets.
- Prepare for loading

MongoDB will be used. Using Python and PyMongo,

- Create Database
- Load data to Collections
- Verify Load success using Query
- Verify Data Integrity

Visualize: -

- Using Matplotlib
- Generate plots that answers our queries

Publish: -

- With HTML & CSS, publish findings on Internet
- Flask (TBD)

Data Extraction

TMDB data extraction

- CSV file is obtained from Kaggle ([link](#)) and locally stored
- We will use budget, revenue, release date and audience rating
- Only movies released from January, 2000 is only considered

OMDB data extraction

- OMDB : JSON data obtained through API request from [OMDB API site](#)
- For each movie in TMDB, an OMDB record is extracted via API request
- Key information we use from OMDB are Metascore (critics rating) & Genre

Data Transformation

Filtering:

Only required columns that will used in current and future analysis will be filtered & transferred to DB

Missing Values:

- Primary columns(Title and Year) - If any of these is null, row is dropped
- Other columns - NA is filled up Zero or mean (if numeric), and "Not provided" (if categorical)

Duplicates:

The row is dropped if both title and release date is duplicated. If only title is duplicated, it indicates a remake and retained

Merging:

TMDB and OMDB are merged by Title and Year columns with inner type of join

All cleaned datasets are saved in CSV and loaded to Mongo DB as well

Loading to Mongo

Reason for Mongo:

We chose Mongo for it

- Ease of use
- flexibility to change columns without structure alteration like mysql

Loading:

- Initialization: Using PyMongo, connection is initialised to localhost. Three collections are created (TMDB cleaned, OMDB cleaned and Merged)
- Loading: Dataframe converted into dictionary is loaded using InsertMany()

Load verification:

- Check #1: verified through the count of records inserted
- Check #2: By querying for a random record and matching it up with DataFrame
- Check #3: By using Mongo Terminal to verify success

Visualization & Publishing

Visualization:

We will be producing plots to analyse,

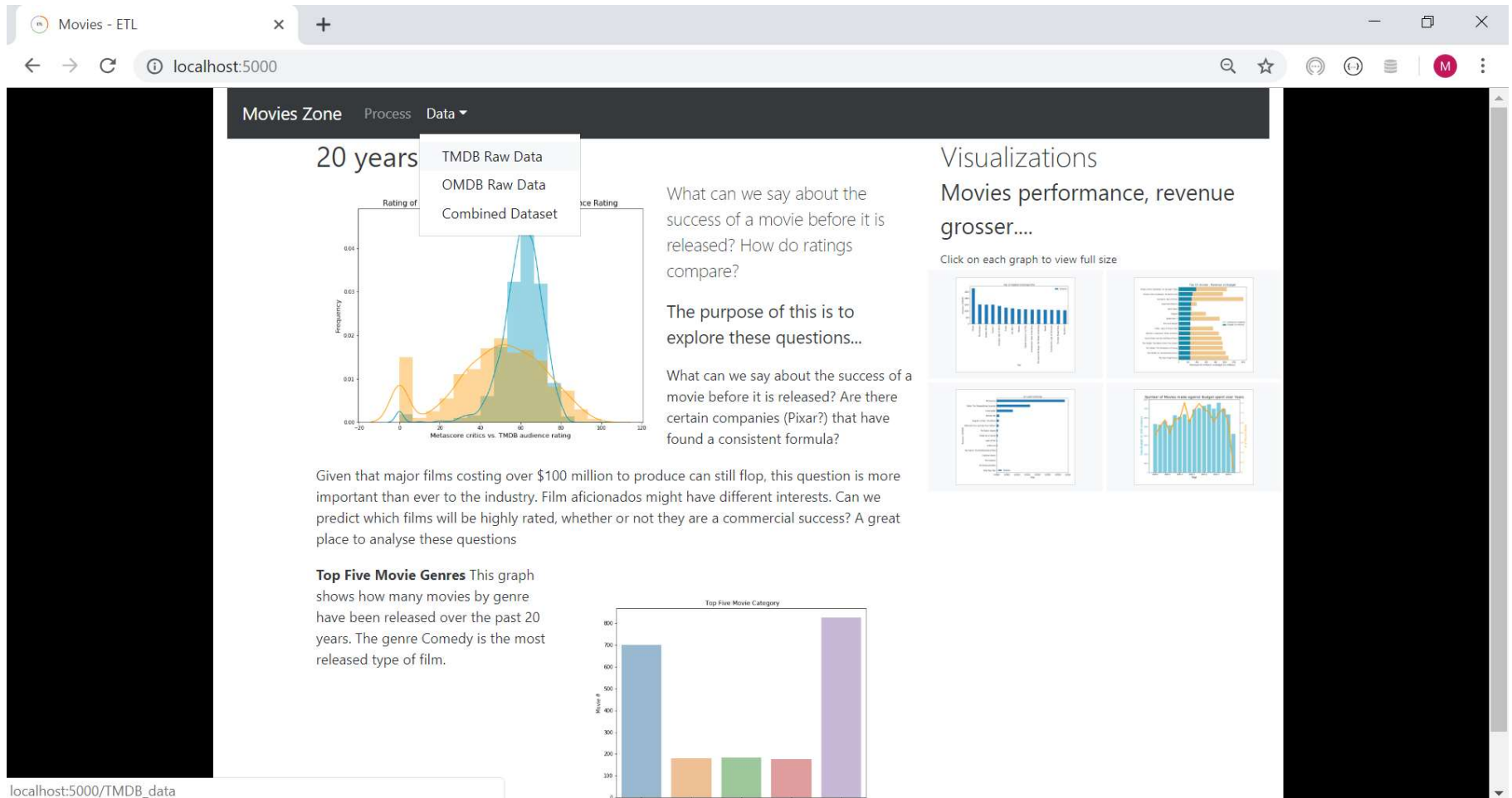
- Top 5 Movie Genres by # of Movies produced
- Revenue: Top 15 and Least 15
- Comaprisn of Budget vs Revenue for Top 15
- Year-wise comaprisn of Budget vs # of Movies Produced
- Comaprisn of Metascore (critics rating) vs Audience Rating

Publishing:

The analysis and data will be published using HTML CSS and rendered through Flask.

Top

# Drop downs to view data page



# Cleaned TMDb source data via Flask

Movies - Data - TMDb

localhost:5000/TMDB\_data

Movies Zone

Process

Data

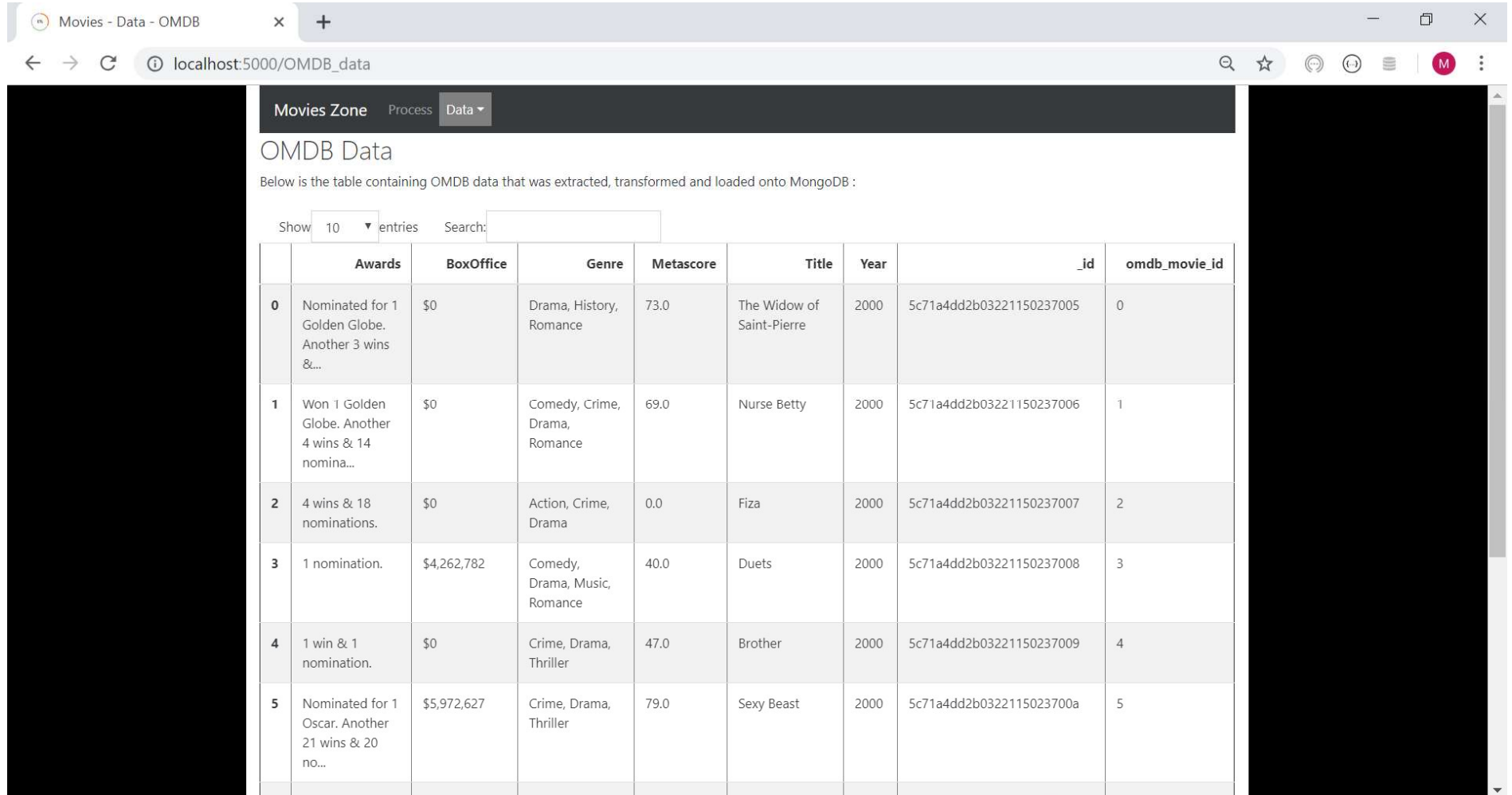
### TMDB Data

Below is the table containing TMDB data that was extracted, transformed and loaded onto MongoDB :

Show 10 entries Search:

	Title	Year	_id	budget	popularity	production_companies	release_date	revenue	runtime	s
0	The Widow of Saint-Pierre	2000	5c71a4de2b03221150237d1f	0	1.780065	[{"name": "Cin\u00e9maginaire Inc.", "id": 280...}	2000-01-01	0	112.00	Rele
1	Next Friday	2000	5c71a4de2b03221150237d20	11000000	9.337388	[{"name": "New Line Cinema", "id": 12}]	2000-01-12	59827328	98.00	Rele
2	My Dog Skip	2000	5c71a4de2b03221150237d21	7000000	5.675535	[{"name": "Alcon Entertainment", "id": 1088}, ...	2000-01-14	0	95.00	Rele
3	Supernova	2000	5c71a4de2b03221150237d22	90000000	5.762037	[{"name": "United Artists", "id": 60}, {"name": ...	2000-01-14	14828081	91.00	Rele
4	Chuck & Buck	2000	5c71a4de2b03221150237d23	0	0.812855	[]	2000-01-21	0	96.00	Rele
5	Down to You	2000	5c71a4de2b03221150237d24	9000000	3.269459	[{"name": "Miramax Films", "id": 14}]	2000-01-21	0	91.00	Rele
6	Urbania	2000	5c71a4de2b03221150237d25	0	0.065155	[]	2000-01-24	0	103.00	Rele
7	Isn't She	2000	5c71a4de2b03221150237d26	36000000	1.068453	[{"name": "Universal	2000-01-28	3003296	95.00	Rele

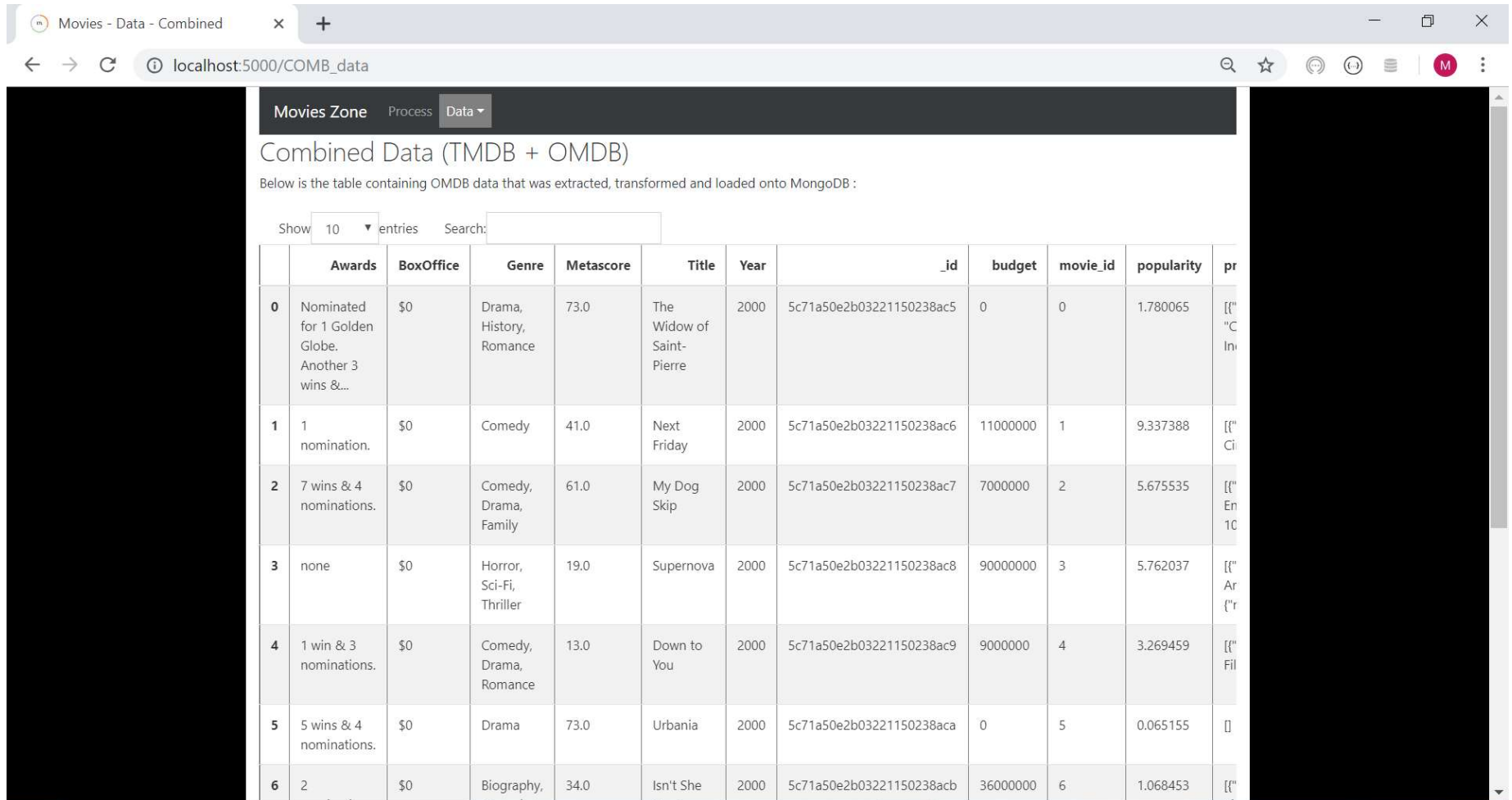
# Cleaned OMDB API data via Flask



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/OMDB\_data'. The page has a dark header with 'Movies Zone' and a 'Data' dropdown menu. Below the header, the title 'OMDB Data' is followed by a description: 'Below is the table containing OMDB data that was extracted, transformed and loaded onto MongoDB :'. A control bar shows 'Show 10 entries' and a search input. The main content is a table with 9 columns: an index column, Awards, BoxOffice, Genre, Metascore, Title, Year, \_id, and omdb\_movie\_id. The table lists five movies from 2000, including 'The Widow of Saint-Pierre', 'Nurse Betty', 'Fiza', 'Duets', and 'Brother'. The table is partially obscured by black bars on the left and right sides of the image.

	Awards	BoxOffice	Genre	Metascore	Title	Year	_id	omdb_movie_id
0	Nominated for 1 Golden Globe. Another 3 wins &...	\$0	Drama, History, Romance	73.0	The Widow of Saint-Pierre	2000	5c71a4dd2b03221150237005	0
1	Won 1 Golden Globe. Another 4 wins & 14 nomina...	\$0	Comedy, Crime, Drama, Romance	69.0	Nurse Betty	2000	5c71a4dd2b03221150237006	1
2	4 wins & 18 nominations.	\$0	Action, Crime, Drama	0.0	Fiza	2000	5c71a4dd2b03221150237007	2
3	1 nomination.	\$4,262,782	Comedy, Drama, Music, Romance	40.0	Duets	2000	5c71a4dd2b03221150237008	3
4	1 win & 1 nomination.	\$0	Crime, Drama, Thriller	47.0	Brother	2000	5c71a4dd2b03221150237009	4
5	Nominated for 1 Oscar. Another 21 wins & 20 no...	\$5,972,627	Crime, Drama, Thriller	79.0	Sexy Beast	2000	5c71a4dd2b0322115023700a	5

# TMDB & OMDB merged data used for analysis & plots via Flask



The screenshot shows a web browser window with the address bar at `localhost:5000/COMB_data`. The application has a navigation bar with 'Movies Zone', 'Process', and 'Data' (selected). The main heading is 'Combined Data (TMDB + OMDB)'. Below it, a text block states: 'Below is the table containing OMDB data that was extracted, transformed and loaded onto MongoDB :'. The table has a 'Show 10 entries' control and a search bar. The table contains 7 rows of movie data with columns: Index, Awards, BoxOffice, Genre, Metascore, Title, Year, \_id, budget, movie\_id, popularity, and popularity.

	Awards	BoxOffice	Genre	Metascore	Title	Year	_id	budget	movie_id	popularity	pr
0	Nominated for 1 Golden Globe. Another 3 wins &...	\$0	Drama, History, Romance	73.0	The Widow of Saint-Pierre	2000	5c71a50e2b03221150238ac5	0	0	1.780065	[[{"
1	1 nomination.	\$0	Comedy	41.0	Next Friday	2000	5c71a50e2b03221150238ac6	11000000	1	9.337388	[[{"
2	7 wins & 4 nominations.	\$0	Comedy, Drama, Family	61.0	My Dog Skip	2000	5c71a50e2b03221150238ac7	7000000	2	5.675535	[[{"
3	none	\$0	Horror, Sci-Fi, Thriller	19.0	Supernova	2000	5c71a50e2b03221150238ac8	90000000	3	5.762037	[[{"
4	1 win & 3 nominations.	\$0	Comedy, Drama, Romance	13.0	Down to You	2000	5c71a50e2b03221150238ac9	9000000	4	3.269459	[[{"
5	5 wins & 4 nominations.	\$0	Drama	73.0	Urbania	2000	5c71a50e2b03221150238aca	0	5	0.065155	[[{"
6	2	\$0	Biography,	34.0	Isn't She	2000	5c71a50e2b03221150238acb	36000000	6	1.068453	[[{"